EV355228515

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Data Classification Using Stochastic Key Feature Generation

Inventor(s):
Cong Li
Ji-Rong Wen
Hang Li

ATTORNEY'S DOCKET NO. MS1-1622US

TECHNICAL FIELD

This invention relates to data classification, and more particularly to data classification using stochastic key feature generation.

BACKGROUND

Classification of data into classes or categories is an often-performed function. One example of such data classification is text classification, which can be used to automatically assign natural language texts into a number of predefined thematic categories. An example of such text classification is email classification at a help desk – email messages are received and are automatically classified into predefined categories so that they can be better responded-to by the help desk staff.

However, the performance of current classification systems can be lacking, especially in situations where the data is noisy (e.g., where the data includes portion(s) that are not useful for classification). Thus, it would be beneficial to improve the performance of classification systems, and thus improve how well data is classified by such systems.

SUMMARY

Data classification using stochastic key feature generation is described herein.

In accordance with certain aspects, a set of training data is obtained having associated summaries. The set of training data and associated summaries are used to generate a key feature generation model. Another set of training data having associated categories is also obtained, and the key feature generation model is

used to map this other set of training data to a set of vectors. A data classifier is then trained based on the set of vectors and the associated categories.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the document to reference like components and/or features.

Fig. 1 illustrates an example environment employing data classification using stochastic key feature generation.

Fig. 2 illustrates an example of a training module of Fig. 1 in additional detail.

Fig. 3 is a flowchart illustrating an example process for training a classifier based on the stochastic key feature generation model.

Fig. 4 illustrates an example of a system for classifying data based on the stochastic key feature generation model.

Fig. 5 is a flowchart illustrating an example process for classifying data based on the stochastic key feature generation model

Fig. 6 illustrates an example of a general computer environment which can be used to implement the techniques described herein.

DETAILED DESCRIPTION

Data classification using stochastic key feature generation is described herein. In accordance with certain aspects, a classifier is trained based on two different spaces: training data and corresponding high-level features. For example, for a text classifier, the two different spaces can be training text and

associated summaries. Once trained, the classifier can be used to classify new input data into a class(es).

Fig. 1 illustrates an example environment 100 employing the data classification using stochastic key feature generation. Environment 100 includes a training module 102 which receives one or more training set(s) 104 that are used by training module 102 to generate a trained classifier 106. Each training set 104 includes training data and one or more high-level (or key) features associated with the training data and categories associated with the training data. Training module 102 uses the training data, the high-level (key) features associated with the training data, and the categories associated with the training data to generate trained classifier 106.

After trained classifier 106 is generated, input data 108 is received at classifier 106 for classification. Trained classifier 106 analyzes input data 108 and classifies or categorizes input data 108 into one or more classes or categories. This classification 110 is output by trained classifier 106.

Additional details regarding the generation of trained classifier 106 and the classification of input data 108 by classifier 106 are discussed below.

Fig. 2 illustrates an example of training module 102 of Fig. 1 in additional detail. Training module 102 includes a stochastic key feature generation model training model 132, a training data mapping module 134, and a classifier training module 136. Two training sets 138 and 140 are illustrated in Fig. 2.

Training sets 138 and 140 are used to train training module 102. Each training set 138 and 140 includes training data as well as one or more of associated summaries and associated categories. In Fig. 2, two training sets 138 and 140 are shown – training set 138 includes training data 142 and associated summaries 144,

while training set 140 includes training data 146 and associated categories 148. Although illustrated as two different training sets 138 and 140, it is to be appreciated that the two training sets may be combined into a single training set. Additionally, it is to be appreciated that one or more additional training sets analogous to training sets 138 and/or 140 can be used for training and input to training module 102.

Training data 142 and training data 146 each include multiple pieces of data that are to be used in the training of training module 102. Training data 142 and training data 146 may include the same pieces of data, may include different pieces of data, or may overlap (e.g., some pieces of data may be included in both training data 142 and 146, while other pieces of data are included in only one of training data 142 or 146).

The type of data included in training data 142 or 146 is the same type of data that is to be classified by the classifier when trained (e.g., classifier 106 of Fig. 1). For example, the data type may be text, image, video, audio, combinations thereof (e.g., web page), and so forth. By way of example, if text documents are to be classified, then training data 142 and 146 would include text pieces (e.g., each piece of text being a text document). By way of another example, if web pages were to be classified, then training data 142 and 146 would include web page pieces (e.g., each piece being an entire web page or part of a web page).

A piece of data (e.g., as included in training data 142 or 146) refers to a collection of data of a particular type(s). Each piece of data is typically the same type of collection of data as is desired to be classified by the trained classifier being created by training module 102. For example, if email messages were to be

classified by the trained classifier, then each piece of data would be an email message (a collection of words). By way of another example, if conference papers were to be classified by the trained classifier, then each piece of data would be a conference paper (a collection of words). By way of yet another example, if images were to be classified by the trained classifier, then each piece of data would be an image. By way of still another example, if web pages were to be classified by the trained classifier, then each piece of data would be a web page (e.g., a collection of words and/or images).

Training set 138 also includes associated summaries 144. Associated summaries 144 include high-level (key) features associated with training data 142. The exact nature of these key features will vary based on the type of data in training data 142. For example, if training data 142 were text, then the key features would be keywords associated with the training data, and each piece of training data would have an associated group of one or more keywords. By way of another example, if training data 142 were images, then the key features would be objects (e.g., people, buildings, scenery, etc.) associated with the training data, and each piece of training data would have an associated group of one or more objects.

Associated summaries 144 can be generated in any of a variety of manners. In certain embodiments, summaries 144 are generated manually (e.g., by having a human user(s) analyze the pieces of training data 142 and identify the key features of each piece to be included in the summary associated with that piece). Alternatively, summaries 144 can be generated automatically. A discussion of example methods that can be used to automatically generate summaries can be found in Dragomir Radev, Simone Teufel Horacio Saggion, Wai Lam, John

Blitzer, Arda Celebi, Hong Qi, Elliott Drabek, and Danyu Liu, "Evaluation of Text Summarization in a Cross-lingual Information Retrieval Framework", Summer 2001 Johns Hopkins Workshop Final Report.

It should be noted that a key feature included in one of the summaries 144 need not be present in the associated piece of data of training data 142. For example, if training data 142 were to be electronic mail (email) messages, and the key features were to be keywords, then a key feature for one email message may be the word "body" even though the word "body" is not present in the associated email message.

Training set 140 also includes associated categories 148. Associated categories 148 include a category or classification assigned to each piece of training data 146. Each piece of training data 146 is assigned to one or more of categories 148. The specific categories or classifications used will be those categories or classifications that the resultant trained classifier is to classify newly input data into. Any of a variety of categories or classifications can be used as desired by the designer of training module 102. In certain embodiments, the categories or classifications are manually defined (e.g., by having a human user(s) determine which categories or classifications are desirable). Alternatively, the categories or classifications can be generated automatically.

By way of example, assume that the designer desires to train a classifier to classify electronic mail (email) messages received at a help desk. The designer can define which categories he or she desires to have the email messages classified into (e.g., a set of common errors or problems encountered by users and their solutions). Multiple email messages are used as training data 146 and are assigned to particular ones of these defined categories (each of the associated categories

148 is one of these defined categories). Additionally, multiple email messages are used as training data 142 (these may be the same email messages as were used as training data 146, or alternatively may be different email messages). A summary for each of the email messages in training data 142 is included in associated summaries 144. Each of these summaries is a list of keywords for the associated email message.

Continuing with the discussion of training module 102 of Fig. 2, stochastic key feature generation model training module 132 trains a stochastic key feature generation model based on the training data 142 and key features collected from the summaries 144. The stochastic key feature generation model denotes the space of conditional probability values in key features. Once this model is trained, given any incoming data, the conditional probability of whether a key feature would occur or would not occur in a summary of the incoming data can be determined.

Stochastic key feature generation model training module 132 trains the stochastic key feature generation model by first obtaining a list of key features. The list of key features can be obtained, for example, by generating a listing of all of the features from the associated summaries 144. Module 132 can also identify, based on summaries 144, for each key feature whether the key feature occurs in the summary of one of the pieces of training data 142. For each key feature, module 132 creates a binary classifier which judges how likely the key feature occurs (or does not occur) in the summary of a particular piece of training data 142.

Reference is made herein to features and key features. Features refer to parts of the input data (e.g., training data, or data to be classified by the trained

classifier as discussed in more detail below). Key features refer to parts of the associated summaries 144 (and which are used in the stochastic key feature generation model training module). For example, in the case of a piece of text, the features would be the words in the piece of text and the key features would be keywords in the summary associated with the piece of text. By way of another example, in the case of an image, the features would be low-level aspects of the image (e.g., colors, textures, etc.) and the key features would be high-level aspects of the image (e.g., people, buildings, scenery, etc.).

The stochastic key feature generation model can also be described using mathematical formulas and equations. Let $\mathcal{X} = \{0,1\}^n$ denote a space of n features and $\mathcal{Y}$ denote a set of classes. Let $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathcal{X}$ denote a feature vector representing a piece of data, in which the i-th element indicates the occurrence or non-occurrence of the i-th feature in the piece of data. Let $y \in \mathcal{Y}$ denote a class representing a category that one or more pieces of data may be classified into. Let $\mathbf{X} = (X_1, X_2, \cdots, X_n)$ denote a random variable on $\mathcal{X}$, and Y a random variable on $\mathcal{Y}$.

Let $\mathcal{S} = \{0,1\}^m$ denote another space of m key features. Let $\mathbf{s} = (s_1, s_2, \cdots, s_m) \in \mathcal{S}$ denote a feature vector representing a summary, in which the i-th element indicates the occurrence or non-occurrence of the i-th keyword in a summary. Let $\mathbf{S} = (S_1, S_2, \cdots, S_m)$ denote a random variable on $\mathcal{S}$.

Given a training set 138 containing training data 142 and associated summaries 144, represented as $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$, stochastic key feature generation model training module 132 constructs a key feature classifier for each of the key features $(S_1, S_2, \cdots, S_m)$. These key feature classifiers are used as probabilistic models in module 132 – that is, the classifiers represent a conditional

probability that a particular key feature accurately represents a particular piece of training data. For example, module 132 can create a naïve Bayesian classifier (*KeyFeatureClassifier)* for each key feature S_j as follows:

$$P(S_j = 1 \mid \mathbf{x}) = \frac{\prod_{i=1}^{n} \hat{P}(X_i = 1 \mid S_j = 1)^{x_i} \hat{P}(S_j = 1)}{\prod_{i=1}^{n} \hat{P}(X_i = 1 \mid S_j = 1)^{x_i} \hat{P}(S_j = 1) + \prod_{i=1}^{n} \hat{P}(X_i = 1 \mid S_j = 0)^{x_i} \hat{P}(S_j = 0)}$$

where $\hat{P}(X_i = 1 \mid S_j = 1)$, $\hat{P}(X_i = 1 \mid S_j = 0)$, $\hat{P}(S_j = 1)$, $\hat{P}(S_j = 0)$ are parameters estimated from the training set 138 using maximum likelihood estimation as follows:

$$\hat{P}(S_j = v) = \frac{count(S_j = v)}{\sum_{u=0,1} count(S_j = u)},$$

$$P(X_i = 1 \mid S_j = v) = \frac{count(X_i = 1 \,\&\, S_j = v)}{\sum_{k=1}^{n} count(X_k = 1 \,\&\, S_j = v)}$$

where *count*() represents the total number of times the condition specified in the parenthetical is satisfied for the particular piece of training data.

Once the *KeyFeatureClassifier* classifiers are created, given a new piece of data $\mathbf{x}$, module 132 can then use the naïve Bayesian key feature classifiers to calculate the conditional probability of each of the keywords: $(P(S_1 = 1 \mid \mathbf{x}), P(S_2 = 1 \mid \mathbf{x}), \cdots, P(S_m = 1 \mid \mathbf{x}))$.

Let Θ denote the space of conditional probability values of key features, and $\boldsymbol{\theta}(\mathbf{x}) = (\theta_1(\mathbf{x}), \theta_2(\mathbf{x}), \cdots, \theta_m(\mathbf{x}))$ a vector in Θ, where $\theta_j(\mathbf{x}) = P(S_j = 1 \mid \mathbf{x})$ $(j = 1, \cdots, m)$. This space Θ, being the collection of vectors $\boldsymbol{\theta}(\mathbf{x})$, is the stochastic key feature generation model created by module 132.

Other types of key feature classifiers could also be used rather than the naïve Bayesian classifier, such as a perceptron classifier or a support vector

machine classifier. Although discussed as using a probabilistic model, stochastic key feature generation model training module 132 could use other models, such as a finite mixture model, statistical machine translation model, and so forth.

The trained stochastic key feature generation model Θ is then output from module 132 to training data mapping module 134. Training data mapping module 134 uses the trained stochastic key feature generation model Θ to map each piece of training data 146 into a probability vector. Each component of the probability vector corresponds to a key feature in the key feature list – the value of each component is the conditional probability of the key feature given the piece of training data 146. These conditional probability values are generated using the *KeyFeatureClassifier* classifiers generated above by module 132. This maps the training data 146 to a vector $\theta(\mathbf{x})$ of the stochastic keyword generation model.

The mapped probability vectors from training data mapping module 134 are input to classifier training module 136, along with the associated categories 148. Each of the mapped probability vectors output by training data mapping module 134 is associated with a piece of training data 146, and thus is also associated with one or more of categories 148. Classifier training module 136 uses a supervised learning algorithm to construct (or train) a classifier h_Θ using the mapped probability vectors from module 134 and their associated categories 148 as follows:

$$h_\Theta(\theta(\mathbf{x})) = \arg\max_y [\sum_{i=1}^{m} w_{i,y}^{(s)} \theta_i(\mathbf{x}) + b_y]$$

where m represents the number of key features (the number of components in the mapped probability vectors), $w_{i,y}^{(s)}$ represents the weight supporting y with the occurrence of the i-th feature in the probability vector, and b_y represents the prior

weight of y. The initial values for these weights can vary based on the supervised learning algorithm used. In certain implementations, using the perceptron algorithm with margins as the supervised learning algorithm, the initial weights are set to zero. The trained classifier $h_{\Theta}(\theta(\mathbf{x}))$ is then output by module 136 as the trained classifier.

Any of a variety of classifiers can be trained by classifier training module 136. Examples of such classifiers include a support vector machine classifier, nearest neighbor classifier, neural network classifier, naïve Bayesian classifier, logistic regression classifier, rule-based classifier, perceptron classifier, and so forth. Each classifier has an associated learning algorithm(s) that is used to train the classifier.

In certain embodiments, classifier training module 136 employs a perceptron algorithm with margins (PAM) as the supervised learning algorithm to train the classifier. The perceptron algorithm with margins can be used to train, for example, a perceptron classifier. An example of the supervised learning algorithm used by module 136 to train the classifier is illustrated by the pseudocode in Table I below. In the example of Table I, the values for τ and η are selected empirically, or alternatively in some other (e.g., ad hoc) manner. As discussed above, the value l corresponds to the size of the training data.

Table I

(Let $\mathbf{w}_y^{(s)}$ denote $(w_{y,1}^{(S)}, w_{y,2}^{(S)}, \cdots, w_{y,m}^{(S)})$)
Select a confidence threshold τ and a learning speed η
for ($y \in \Upsilon$) $\mathbf{w}_y^{(s)}(0) \leftarrow 0, b_y(0) \leftarrow 0$
$t \leftarrow 0, R = \max_{\mathbf{x}_i} \|\boldsymbol{\theta}(\mathbf{x}_i)\|$
repeat
  for $i = 1,2,\cdots,l$
    for $y \in \Upsilon$
      if $y = y_i$ **then** $z \leftarrow 1$
          else $z \leftarrow -1$
      if $z(\boldsymbol{\theta}(\mathbf{x}_i) \cdot \mathbf{w}_y^{(s)}(t) + b_y(t)) \le \tau$ **then**
        $\mathbf{w}_y^{(s)}(t+1) \leftarrow \mathbf{w}_y^{(s)}(t) + \eta z \boldsymbol{\theta}(\mathbf{x}_i)$
        $b_{t+1} \leftarrow b_{t+1} + \eta z R^2$
        $t \leftarrow t+1$
      end if
    end for
  end for
until no updates made within the **for** loop

Fig. 3 is a flowchart illustrating an example process 180 for training a classifier based on the stochastic key feature generation model. The process of Fig. 3 is performed, for example, by training module 102 of Figs. 1 and 2, and may be implemented, for example, in software, firmware, hardware, or combinations thereof.

Initially, training data and associated key features are obtained (act 182). The training data and associated key features can be different types, as discussed above, such as pieces of text and keywords associated with those pieces. The obtained training data and associated key features are used to generate a stochastic key feature generation model (act 184). As discussed above, the stochastic key feature generation model denotes the space of conditional probability values in key features.

Training data and associated categories (or classes) are then obtained (act 186). The training data in act 186 may be the same (or partially the same) as the training data in act 182, or alternatively may be different. The training data obtained in act 186 is then mapped to vectors using the stochastic key feature generation model generated in act 184 (act 188). The resultant vectors are thus also associated with the categories obtained in act 186.

A classifier is then trained using the mapped vectors from act 188 and the categories associated with those vectors from act 186 (act 190). The resultant trained classifier from act 190 can then be used to classify or categorize data input to the classifier.

Fig. 4 illustrates an example of system 200 for classifying data based on the stochastic key feature generation model. System 200 includes a stochastic key feature generation model-based vector generation module 202 and a classifier 204.

Stochastic key feature generation model-based vector generation module 202 receives as an input a stochastic key feature generation model 206 and data 208. The stochastic key feature generation model 206 is the model trained and output by stochastic key feature generation model training module 132 of Fig. 2. The data 208 is the data that is to be classified.

Stochastic key feature generation model-based vector generation module 202 uses the trained stochastic key feature generation model Θ to map data 208 into a vector. Module 202 operates in the same manner as training data mapping module 134, except that module 202 is operating on data to be classified rather than training data. This maps the data 208 to a vector $\theta(x)$ of the stochastic keyword generation model.

The vector generated by module 202 is then input to classifier 204. Classifier 204 is the trained classifier output by classifier training module 136 of Fig. 2. Classifier 204 uses the vector received from module 202 to classify the data 208 into one or more classes or categories.

It should be noted that system 200 does not require summaries for data 208 in order for data 208 to be classified. If a summary for data 208 exists, then the summary can optionally be included as part of the data input to module 202, or alternatively the summary can be ignored by module 202.

Fig. 5 is a flowchart illustrating an example process 230 for classifying data based on the stochastic key feature generation model. The process of Fig. 5 is performed, for example, by system 200 of Fig. 4, and may be implemented, for example, in software, firmware, hardware, or combinations thereof.

The data to be classified is received (act 232), and the stochastic key feature generation model is used to obtain a vector representing the data (act 234). Once the vector is obtained, the trained classifier is used to obtain a classification or category or class for the vector (act 236). As the vector is based on the received data, this classification or category or class obtained in act 236 is also the classification or category or class for the data.

Fig. 6 illustrates an example of a general computer environment 300, which can be used to implement the techniques described herein. The computer environment 300 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computer environment 300 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the example computer environment 300.

Computer environment 300 includes a general-purpose computing device in the form of a computer 302. Computer 302 can implement, for example, training module 102 of Fig. 1 or 2, or system 200 of Fig. 4. The components of computer 302 can include, but are not limited to, one or more processors or processing units 304, a system memory 306, and a system bus 308 that couples various system components including the processor 304 to the system memory 306.

The system bus 308 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, and a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus.

Computer 302 typically includes a variety of computer readable media. Such media can be any available media that is accessible by computer 302 and includes both volatile and non-volatile media, removable and non-removable media.

The system memory 306 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 310, and/or non-volatile memory, such as read only memory (ROM) 312. A basic input/output system (BIOS) 314, containing the basic routines that help to transfer information between elements within computer 302, such as during start-up, is stored in ROM 312. RAM 310 typically contains data and/or program modules that are immediately accessible to and/or presently operated on by the processing unit 304.

Computer 302 may also include other removable/non-removable, volatile/non-volatile computer storage media. By way of example, Fig. 6 illustrates a hard disk drive 316 for reading from and writing to a non-removable, non-volatile magnetic media (not shown), a magnetic disk drive 318 for reading from and writing to a removable, non-volatile magnetic disk 320 (e.g., a "floppy disk"), and an optical disk drive 322 for reading from and/or writing to a removable, non-volatile optical disk 324 such as a CD-ROM, DVD-ROM, or other optical media. The hard disk drive 316, magnetic disk drive 318, and optical disk drive 322 are each connected to the system bus 308 by one or more data media interfaces 326. Alternatively, the hard disk drive 316, magnetic disk drive 318, and optical disk drive 322 can be connected to the system bus 308 by one or more interfaces (not shown).

The disk drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules, and other data for computer 302. Although the example illustrates a hard disk 316, a removable magnetic disk 320, and a removable optical disk 324, it is to be appreciated that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random access memories (RAM), read only memories (ROM), electrically erasable programmable read-only memory (EEPROM), and the like, can also be utilized to implement the example computing system and environment.

Any number of program modules can be stored on the hard disk 316, magnetic disk 320, optical disk 324, ROM 312, and/or RAM 310, including by

way of example, an operating system 326, one or more application programs 328, other program modules 330, and program data 332. Each of such operating system 326, one or more application programs 328, other program modules 330, and program data 332 (or some combination thereof) may implement all or part of the resident components that support the distributed file system.

A user can enter commands and information into computer 302 via input devices such as a keyboard 334 and a pointing device 336 (e.g., a "mouse"). Other input devices 338 (not shown specifically) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and other input devices are connected to the processing unit 304 via input/output interfaces 340 that are coupled to the system bus 308, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

A monitor 342 or other type of display device can also be connected to the system bus 308 via an interface, such as a video adapter 344. In addition to the monitor 342, other output peripheral devices can include components such as speakers (not shown) and a printer 346 which can be connected to computer 302 via the input/output interfaces 340.

Computer 302 can operate in a networked environment using logical connections to one or more remote computers, such as a remote computing device 348. By way of example, the remote computing device 348 can be a personal computer, portable computer, a server, a router, a network computer, a peer device or other common network node, and the like. The remote computing device 348 is illustrated as a portable computer that can include many or all of the elements and features described herein relative to computer 302.

Logical connections between computer 302 and the remote computer 348 are depicted as a local area network (LAN) 350 and a general wide area network (WAN) 352. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When implemented in a LAN networking environment, the computer 302 is connected to a local network 350 via a network interface or adapter 354. When implemented in a WAN networking environment, the computer 302 typically includes a modem 356 or other means for establishing communications over the wide network 352. The modem 356, which can be internal or external to computer 302, can be connected to the system bus 308 via the input/output interfaces 340 or other appropriate mechanisms. It is to be appreciated that the illustrated network connections are examples and that other means of establishing communication link(s) between the computers 302 and 348 can be employed.

In a networked environment, such as that illustrated with computing environment 300, program modules depicted relative to the computer 302, or portions thereof, may be stored in a remote memory storage device. By way of example, remote application programs 358 reside on a memory device of remote computer 348. For purposes of illustration, application programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computing device 302, and are executed by the data processor(s) of the computer.

Various modules and techniques may be described herein in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include

routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

An implementation of these modules and techniques may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise "computer storage media" and "communications media."

"Computer storage media" includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer.

"Communication media" typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or

direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

Various flowcharts are described herein and illustrated in the accompanying Figures. The ordering of acts in these flowcharts are examples only – these orderings can be changed so that the acts are performed in different orders and/or concurrently.

Although the description above uses language that is specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the invention.